430-TP-002-001

PDPS Data Processing Prototype Phase 1 Report

Technical Paper

Technical Paper--Not intended for formal review or government approval.

November 1994

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

J. Martin /s/ 12/14/94

Jolyon Martin, PDPS Scientist / Engineer Date
EOSDIS Core System Project

SUBMITTED BY

Parag N. Ambardekar 12/14/94

Parag Ambardekar, PDPS Manager Date
EOSDIS Core System Project

Hughes Applied Information Systems
Landover, Maryland

This page intentionally left blank.

Abstract

We have successfully achieved the targets for the first phase of the data processing prototype. This development represents a generic, stand-alone processing environment using the SDS design concepts for data processing. The prototype has been demonstrated using real processing algorithms from the pathfinder activities both as heritage algorithms and as encapsulated within the SDP Toolkit. The primary purpose of this report is to evaluate the prototype after the first phase and discuss issues raised within the work.

Keywords: PDPS, Data Processing, Prototype, Queuing

This page intentionally left blank.

Contents

Abstract

1. Introduction

2. Software Overview

3. Hardware Overview

4. Chains Interface Evaluation

5. Queuing Services Evaluation

6. Prototype Issues

References

This page intentionally left blank.

1. Introduction

1.1 Overview

This report presents the results and issues arising from the first phase of the data processing prototype. The broad aims for the data processing prototype have been advertised as follows:

- to verify SDS concept
- to furnish requirements / design information
- to support SDP toolkit development
- to provide early opportunity for community feedback
- to support end-to-end ‘push’-side prototyping (longer term)

The prototype has been split into phases to allow an incremental approach to the development with the opportunity for review and redirection after each phase. The planning of the prototype was originally split into two phases, a proof of concept, followed by an iteration, intended to support any ECS ‘end-to-end prototyping’ activities. In order to better interface with the overall ECS project schedule, and the schedule of other prototype components that could collaborate in an end-to-end prototype, a three phased approach is now envisaged:

1.1.1 Phase 1: Proof of Concept

Scope

- manage, queue, execute data processing requests at a provider site.
- support process monitoring / control by operations staff

Drivers

- generate a generic solution for a data processing environment
- configurable to any processing chain
- provide standalone environment by integrating (C)OTS and heritage software.

Schedule

- 8/94 to 11/94

1.1.2 Phase 2: Data Processing Issues

Scope

- The proposed targets of this phase are to investigate a number of technical issues arising from the prototype (identified in chapter 6). The output from this phase will provide detailed information for the PDR activities.

Schedule

- 12/94 to 2/95

1.1.3 Phase 3: Iteration for End-to-End Prototype

Scope

- The proposed targets of this phase are to address the interfaces required to support an end-to-end prototype in collaboration with other prototype activities, e.g. planning and data server prototypes if possible.

Schedule

- 3/95 to 5/95

We have successfully achieved the targets for the first phase in developing a generic, stand-alone processing environment using the SDS design concepts for data processing. The prototype has been demonstrated using real processing algorithms from the pathfinder activities [ref. 1,2] (both as heritage algorithms and using the SDP Toolkit). The output of this first phase is being used to feed PDPS requirements and design as well as providing a base level system to be used in the future phases of prototype.

The primary purpose of this report is to evaluate the prototype after the first phase and discuss issues raised within the work.

1.2 Organization

As a guide to the rest of this report:

Chapter 2: Software Overview, presents an overview of the software components of the prototype and the mapping of these components to the SDS service classes. The information here summarizes the more detailed design notes presented in the prototype preliminary design document [ref. 8].

Chapter 3: Hardware Overview, presents an overview of the hardware configuration used for the prototype.

Chapter 4: Chains Interface Evaluation, discusses the chains GUI interface and observations made during preliminary demonstrations. The further development and re-use of this component are also discussed.

Chapter 5: Queuing Service Evaluation, discusses the queuing services utilized in the prototype. The experience of having worked with DQS within the prototype is used to generate some evaluation criteria for this category of services.

Chapter 6: Prototype Issues, discusses various issues that have been raised during the prototype. The present approach to the issues within the prototype is discussed, and follow-up plans are presented

1.3 Acknowledgments

The Authors of this report; Jolyon Martin and Simon Tackley, would especially like to thank the following people for their assistance during the first phase of prototype:

- Narayan Prasad and Merik Chmielowski (ECS) for providing SDP Toolkit versions of science algorithms for demonstrating and testing the data processing prototype.
- Mary James and the AVHRR Pathfinder Team (GSFC) for providing operations information for the AVHRR Pathfinder algorithm [ref. 3].
- DQS development staff from Florida State University for support of Distributed Queuing System.

Questions regarding technical information contained within this Paper should be addressed to the following ECS contacts:

- ECS Contacts
  - Jolyon Martin, PDPS Scientist / Engineer, (301) 925 0436, jolyonm@eos.hitc.com
  - Simon Tackley, PDPS Scientist / Engineer, (301) 925 0462, stackley@eos.hitc.com

2. Software Overview

This section presents an overview of the software within the Data Processing subsystem Phase 1 prototype.

2.1 Design Overview

The software components of the phase 1 prototype have been generated using both COTS and Heritage code. Heritage libraries developed by EOSL for processing control applications have been used to develop an application called CHAINS. This forms the front end of the system, allowing entry of processing request data and monitoring of process execution through a GUI. The CHAINS application interfaces to a distributed queuing package. The queuing package used in the prototype is the Distributed Queuing System DQS [ref. 4]. This package comprises of two basic services to a) manage and queue processing requests (DQS Master) and b) execute them on relevant machines (DQS Exec). This is summarized in Figure 1.

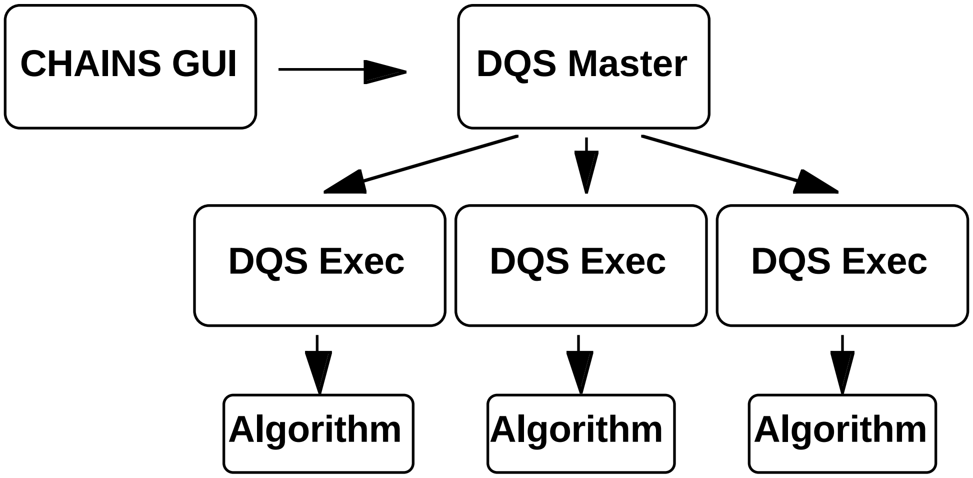


Figure 1: Software Components Overview

The mapping of software to the SDS Service Classes is described in detail in the preliminary design documentation for the prototype [ref. 8]. The mapping is summarized in Figure 2.

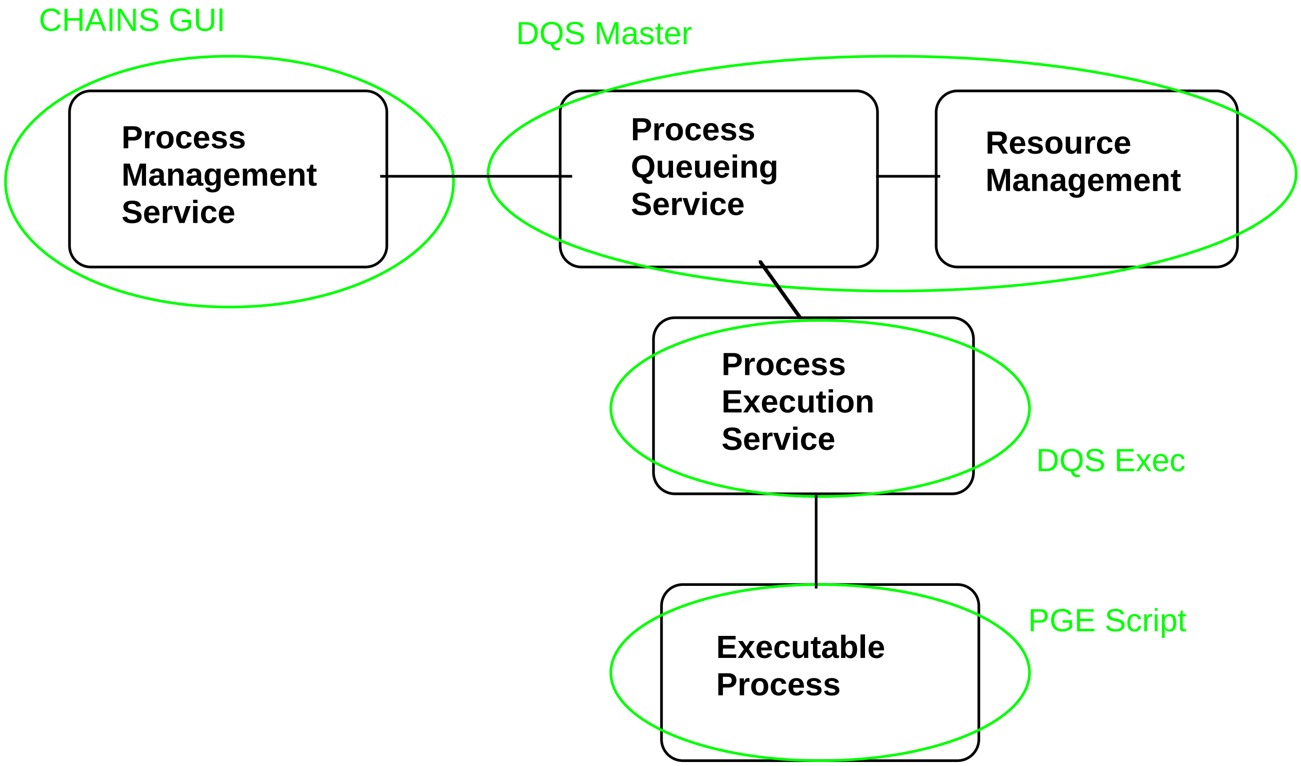


Figure 2: Service Class mapping for the Data Processing Prototype

Detailed design for the data processing environment is contained in on-line design documentation[1]. The following selections provide a summary of the components in the prototype.

[1] An HTML preliminary design document is accessible from the SDPS Prototype page within the EDHS WWW document server.

2.2 PGE scripts

The basic unit which is queued and executed in the prototype is a PGE script. This is very typical for most queuing packages. Some queuing systems require the script to be specified in a given language (e.g. JCL in Control-M). DQS allows most standard script languages to be used. Directives for the queuing services are frequently specified within the script (comment lines containing resource requirements are an example of this within DQS).

The PGE scripts have to contain the control information necessary for the PGEs to execute - i.e. the Process Control information as accessed by the SDP Toolkit, together with any environmental variables etc. The PGE scripts are generated within the CHAINS software upon reception of a Data Processing Request and sent to DQS for execution. The responsibility of the process management service in the prototype is to take the processing parameters etc. which are part of the data processing request, and to use these to qualify a template script. A template script is set up for each processing algorithm. In this first phase of prototype the Data Processing Requests are entered manually via a GUI (to qualify the template script).

Using a script as the scheduled unit has been found to be a very flexible mechanism; pre-processing, post processing, error recovery steps can be included in the script. As the operations of the processing become better understood the processing scripts can be expanded with ease.

2.3 EOSL CHAINS

The EOSL CHAINS software consists of a set of libraries for controlling and monitoring algorithm execution. Interface is via a GUI. Effort has gone in during the Phase 1 prototyping to provide API interfaces also, both for portability reasons and to support the longer term objective of an end-to-end infrastructure prototype. Within the Phase 1 prototype it performs the roles of

- generating Data Processing Requests (in place of Planning subsystem)
- handling and processing Data Processing Requests as though it were the Process Management Service Class
- monitoring the execution of resulting PGEs.

The CHAINS software handles chains of PGEs. An algorithm chain is comprised of a set of 1 or more PGEs. Chain types are defined in simple ASCII templates. The software allows loading and execution of multiple chain types.

In terms of content, the CHAINS software consists internally of a number of libraries, namely:

- Finite State machine library
- Local process management library
- Queued process management library
- Display library
- Run library

- Script editor library
- Mime library
- Support tools library.

The finite state machine library handles the status of the various elements of the chain. The PGEs can currently be processed locally or via the distributed queuing system; the chain execution logic is different in the two cases. The local process management module and the queued process management module handle the submission of PGEs for execution. The interface between the CHAINS software and the queuing system is through high level interfaces via a service layer, hence reducing coupling between the packages and allowing easy substitution of queuing packages at a later date.

The display library creates a graphic representation of the algorithm chain, with each PGE visualized as a datastore, a controller and a PGE object. The controller is in the visual form of a traffic light, and allows the operator to start, hold, release and cancel the PGE. The default presentation of the chain is as a linear chain of PGEs; however it is possible to view the dependencies within a chain by moving to a split view, which positions the graphical objects relevant to their dependencies. Basic status information (Unstarted / Queued / Executing / Completed / Error) is shown for all PGEs.

The management of chains of PGEs is handled by the run library, which controls their loading and causes their display by the display module.

The processing scripts which encapsulate PGEs are created from script templates, using a value-substitution tool, the Script Editor. These scripts contain information such as data filenames and directories, processing parameters, log file locations etc. A script template is an ASCII text containing variable markers, which are replaced with the values received in the data processing requests. These templates are currently seen as an output of the AI&T process. The Script Editor module provides both API and GUI interfaces, to allow automation in future phases of prototyping. Facilities are provided for the Operator to be made aware of unqualified values within the scripts and to edit and view values as part of both the data processing request entry and validation processes. The Script Editor is designed in such a way as to enable easy re-use.

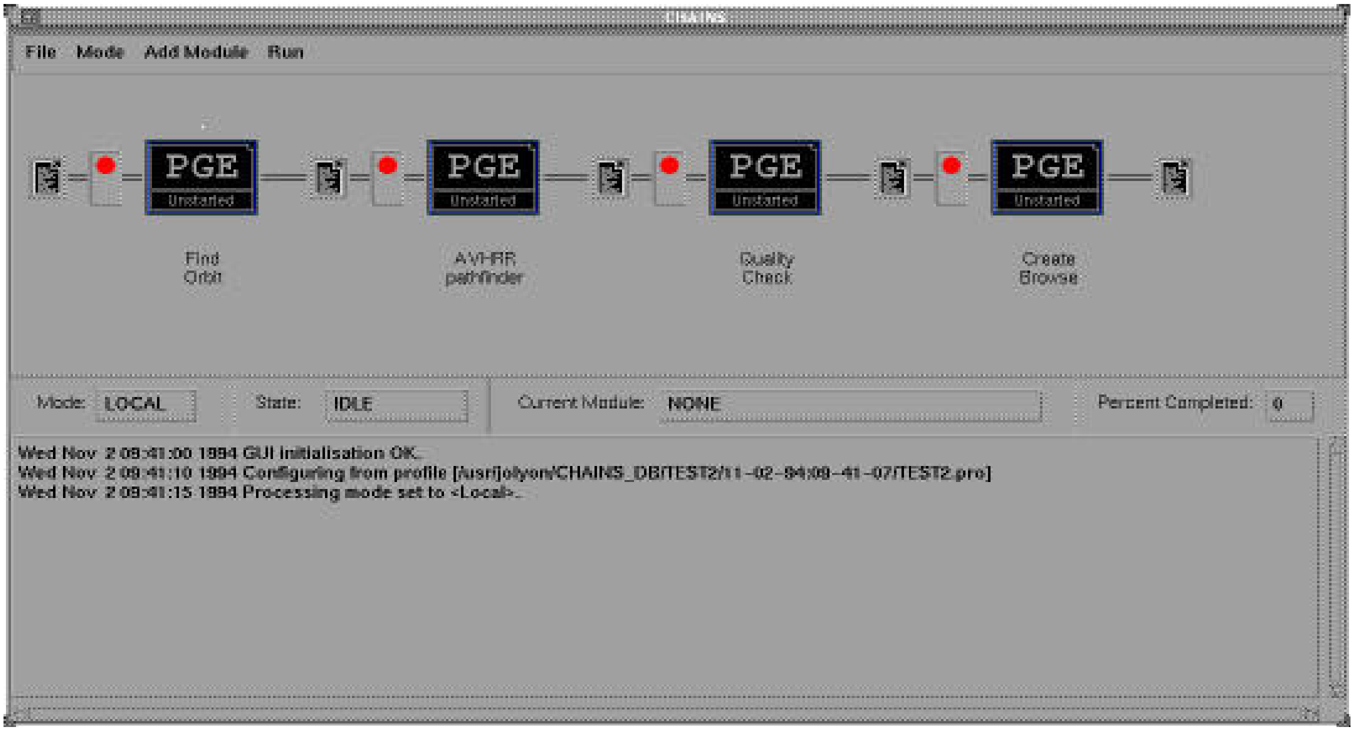
Facilities for viewing PGE input and output data are included. This function utilizes the mime library, which automatically matches viewers to file types based on file extensions, using the same type of technology applied by more advanced E-mail tools and the Mosaic internet tool.

The mime library is also utilized within the Support tools module for viewing Help files. Help files are currently attached both to the CHAINS software and to individual PGEs. A log file is also attached to the CHAINS interface and may be viewed and printed. A queue monitoring tool is included as part of the Support tools module, as are tools for viewing and editing the algorithm processing scripts directly. Information about old runs may be loaded, to allow the operator to view information from previous operator sessions.

Each of these libraries is described in more detail in the design documentation.

2.4 GUI Overview

The front end interface to the prototype is provided by the CHAINS software. As described in the software overview this interface has been developed from a set of heritage libraries (originally used to support graphical configuration of processing SAR data). A snapshot view of the GUI is shown below, however the GUI interface is best seen “in real life” rather than described in this report.



Mode: LOCAL State: IDLE Current Module: NONE Percent Completed: 0

```
Wed Nov  2 09:41:00 1994 GUI initialisation OK.
Wed Nov  2 09:41:10 1994 Configuring from profile [/usr/jolyon/CHAINS_DB/TEST2/11-02-94:09-41-07/TEST2.pro]
Wed Nov  2 09:41:15 1994 Processing mode set to <Local>.
```

Figure 3. Snapshot of the CHAINS GUI

The following list summarizes some of the features presented by the GUI, and the capabilities that these features provide to operators monitoring and controlling the processing.

- Generic configuration of multiple run types: templates are created to describe chains of processing, with complex dependencies between steps (PGEs) in the chain:
- Local or Distributed (Queued) Processing: the choice of processing locally to the host of the GUI or distributed within a heterogeneous clustered workstation environment
- Graphical Representation of PGE: status monitoring of the PGE as an icon within the chain, with the capabilities for the operator view / manipulate the associated data objects with the PGE, for example:

- PGE Help - to view associated help documentation of the PGE
- Set Parameters - an interface to set processing parameters for the PGE
- Edit/View Script - the ability to edit / view the processing script of a PGE
- View Input/Output Data - to view the various data products of the PGE
- View Log - to view the processing log for the PGE
- Skip Module - to miss a step in the processing chain
- Traffic Lights - to control the data flow within the processing chain

- Status Monitoring: Processing log records the events within the data processing environment, dynamic display of the allocation of processing requests to resources and pending queued data processing requests.

For a more complete description of the features of the GUI see the on-line help page.

2.5 DQS Queuing Software

The DQS package comprises a manager service, DQS Master, which handles accepting and distributing processes, and an execution service DQS Exec, an instantiation of which runs on each processing resource.

The submission of jobs to processing resources managed by DQS is based on resource matching and load balancing. DQS has a mechanism known as queue complexes whereby abstract attributes are attached to a resource. The semantics of these attributes are fairly simple; there are three possibilities:

attribute

attribute = string

attribute = value

These are used to express the capabilities of a resource: e.g.

architecture = sgi

memory = 64

disk = 1000

pvm

matlab

Then jobs are dispatched to the relevant machine based on their specification of resource requirements e.g.

architecture.eq.hp

memory.gt.32

disk.gt.100

pvm

A simple load balancing mechanism is used to ensure that machines are utilized according to their processing loads. A five minute average of the processing load (as measured by % utilization of the processor capability) is used to balance processing loads. The software performing the load averaging is based on software provided by the GNU foundation.

3. Hardware Overview

The Data Processing subsystem will support a heterogeneous processing environment, and hence the DP prototype has been developed in a similar environment.

A small number of distributed workstations were set up to be managed within DQS. Three different UNIX platforms have been included to verify heterogeneous operation. The hardware is illustrated in Figure 4.

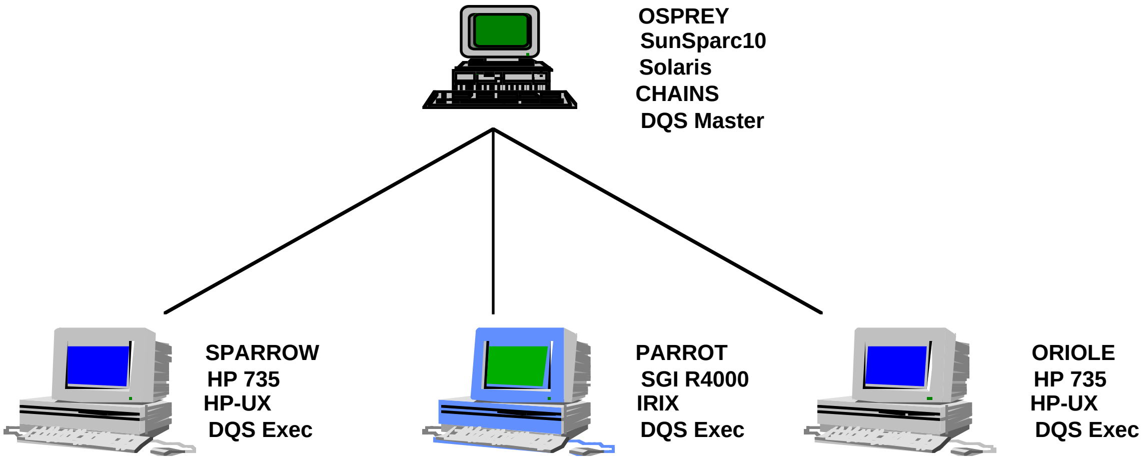


Figure 4. Hardware Overview

The DQS Master and CHAINS interface were installed onto a machine named "osprey", and three workstations set up a processing resources ("sparrow", "oriole", "parrot"). The machine named "parrot" was given a "QA" attribute to verify the use of arbitrary attributes in determining the allocation of processing resources.

The choice of hardware used in the prototype was entirely arbitrary and should not be assigned any significance, beyond its heterogeneous nature. The CHAINS software was installed on "osprey" since the MOTIF graphics libraries and development environment required were available on this machine. The DQS master process was installed on the same machine because of its close interface with the CHAINS software.

The DQS package has been ported to a large number of UNIX platforms, hence allowing a widely heterogeneous processing environment to be supported. Within the ECS workstation environment DQS has been built and run on the following classes of platform:

- DEC, IBM, SGI, SUN (solaris and sun-os 4.x), HP

4. Chains Interface Evaluation

4.1 Evaluation

Preliminary demonstrations of the GUI have identified that the overall CHAINS approach is considered as ideal during the early stages of algorithm operations, especially during AI&T and operational testing. The general response to the prototype has been very positive, the ability to view the progress of data processing chains, manage the control flow, view intermediate data from the GUI are considered essential functions for routine operations.

The evaluations thus far have identified a number of features that should be made available (either in the final system design or in iterations of the prototype):

An overall summary display of all the runs at the processing site would be a necessary addition for monitoring the routine operations at a DAAC, (with the ability to navigate from the summarized run to the CHAIN view currently implemented for error investigation).

Another point for consideration is the alternate views on the processing given from the planning sub-system and from the aspect of resource management. It is not entirely clear how these will compliment / supplement the CHAINS type display. The planning sub-system will contain a view of the active plan of data processing, and the status of the processing against that plan. This is typically a higher level summary view of the data processing activities. The resource management subsystem will typically contain a GUI to all the resources at a processing site, and show the load of processing on the resources. The CHAINS display is a good model for the capabilities required in the data processing environment, there is however a valid concern as to how this interface would interface with the other views on processing.

A useful enhancement to the CHAINS view that has been suggested is that as well as identifying the PGE’s being used in the chain, a summary of the data would also be useful.

Other early evaluation comments have already been included in the GUI these included

- PGE Help
- Process Log viewing

4.2 Re-use potential

In order to be capable of advancing to support an end-to-end prototype, the operator interface to setting up processing requests, i.e. manually entering processing parameters etc. has to be supplemented by a software interface for the planning sub-system. At this point it would seem most reasonable to split the CHAINS program into an interface component and a true service component. This split was anticipated in the design of the software enhancement in phase 1, and the coupling of the GUI components and service components are designed to minimize the modifications that would be required during this split.

5. Queuing Services Evaluation

There are a large number of queuing systems, available as COTS or freeware, providing similar sets of features and functions. The processing prototype uses the Distributed Queuing System (DQS) to provide queuing services to distribute and manage processing on the workstation cluster. This chapter describes the basic functions of queuing packages, and from the experience of the prototype, presents a candidate list of evaluation criteria for other queuing systems. An evaluation of the applicability of DQS as a solution for the ECS queuing service is also presented.

5.1 Standard Queuing Services

Many of the queuing packages share a common lineage to the Network Queuing System (NQS) [ref. 4]. This package has in some ways defined the fundamental services required / expected from queuing packages. The POSIX standard P1003.2d (presently draft in status) presents an effort to standardize the interfaces to these queuing services [ref. 5]. The POSIX standard defines a number of requests for obtaining services from queuing (or batch) packages. By standardizing the requests in a client, server framework it should be possible for queuing services to interoperate. The general philosophy behind the standard is to permit the vendor to implement the mechanism for managing the processing jobs etc. in the way most suited to their platform.

It remains to be seen whether the various vendors / developers of queuing packages will sign up to any standards. There are candidate packages being developed to the standard, e.g. Portable Batch System PBS [ref. 6], and indications that other packages are using the standard as a guide for newly developed feature, but the goal of interoperability seems to be fairly distant at present.

Within the prototype the interfaces are encapsulated as methods of the PGE object, (queue, hold, release etc.). The precise semantics of the interface are then hidden from the majority of the application code, in this way it should be possible to plug and evaluate with other applications.

5.2 Queue Services Evaluation Criteria

The report “A Comparison of Queuing, Cluster and Distributed Computing Systems” [ref. 4] contains a set of evaluation criteria for this class of packages, not all of these criteria however are applicable for the ECS environment. For example many packages contain features for monitoring user activity on the distributed workstations, and can suspend queued processing when a user (for example) logs into the machines console. This type of feature is not strictly necessary for the ECS environment where resources will generally be dedicated to processing tasks. The following is a list of evaluation criteria is built from experience with the phase 1 prototype, along with some general observations.

5.2.1 Functional Criteria

1 Standard Services: The standard set of services described in the POSIX standard give a good indication of required interfaces for the queuing services. These interfaces are:

qalter - alter the attributes of a job

qdel - delete job

qhold - hold a job

qmove - move a job

qmsg - send message to a job

qrerun - rerun a job

qrls - release a job

qselect - select a job

qsig - signal a job

qstat - show status of job

qsub - submit a job

Queuing packages do not always present these services in the same way, however these POSIX capabilities indicate most of the required functions

2 Job Dependencies: The queuing services for the data processing environment require mechanisms to specify dependencies between processing jobs.

3 Job Priorities: The queuing services for the data processing environment require mechanisms for specifying relative priorities between processing jobs.

4 Job Dispatching: Job dispatching algorithms within the queuing services are typically based on resource matching and load balancing.

a Resource Matching: The queuing services should be able to both specify the resource for processing explicitly as well as matching processing job resource requirements to the available resources in the data processing environment.

b Load Balancing: It is anticipated that the ECS resources will be dedicated to processing of single jobs at a time, however it is possible that resources could be shared between concurrent processes (small ad-hoc jobs for example) and as such queuing services should be aware of the relative loads of the processing resources in order to submit processing jobs to the optimum resources.

5 Job Limits: The queuing services should be able to monitor processing jobs against defined limits of resource usage so that jobs and have capabilities to manage these jobs in a well defined manner when usage exceeds the limits.

6 Job Status: The queuing services should be able to report the status of processing jobs. Typically there are several levels at which the reporting is required, status of the job

within the queuing services (queued, waiting, running etc.) as well as status of the job when processing.

7 Resource Reporting: The queuing services should be able to report the resource usage of the processing job (to provide information for future planning of the processing).

8 Checkpointing: The support for checkpointing the processing jobs should be assessed.

9 Resource Scheduling: The support for resource scheduling within the queuing package should be evaluated (the ability to look forward to prevent jobs being started on platforms which are scheduled for down-time -- this is not a standard feature in most COTS packages).

5.2.2 Software Criteria

10 Heterogeneity: The queuing package will have to run on all the heterogeneous platforms which are required in the ECS.

11 Parallel Support: The level of support for parallel processing paradigms will have to be evaluated if parallel processing paradigms are to be used in the ECS.

12 No Single point of failure: The safety features of the software should be evaluated, typical packages contain no single point of failure, as well as well defined recovery procedures following loss of processing resources etc.

13 Operator interface (GUI): The operator interface to the package is required for administration of the queuing service. The features which are available from a GUI should be evaluated.

14 API interfaces: The provision of different levels of access to the software should be evaluated, three levels of interface are ideal: GUI, command-line and API, although all three of these levels are not always required, depending on other support software in the processing environment.

5.2.3 Other Criteria

There are a number of features that are frequently bundled together within queuing systems to enable fuller management of the production environment. These features (or services) aren’t necessarily required within the queuing system since they may be supplied by the management infrastructure or other components of the ECS. However, these features should be evaluated in addition in order to assess either how well the functions interface to the management functions, or whether the packages functions could in fact supplant the other services.

15 Resource Monitoring: display of the processing resources usage and load on the processing environment is frequently supported in the queuing packages.

16 Reporting: report generation of processing activities is also frequently supported in the data processing packages.

16 Security: Many of the queuing packages contain security management functions, since they often used in open environments that allow many users of a cluster of workstations to submit processing jobs. Security administration is not part of the data processing

requirements per-se since it is managed by the management services, however the applicability of the security interfaces within the package should be matched against those management requirements.

17 Accounting: Similarly to security, accounting often is a part of queuing packages.

18 Planning: The support for future planning of processing should be evaluated.

5.3 DQS Evaluation

5.3.1 Functional Criteria

DQS contains only the core services qsub, qdel, qstat at present. There are plans to implement the other interfaces in later releases. The package supports job dependencies, but not priorities at present (again there are future plans for priorities)

One of the major differences in various queuing packages is the resource management / resource model capabilities of the package. The resource model within a queuing package is used to manage how a job to be scheduled or dispatched onto an appropriate machine. In the SDS the objects used to describe the resource matching were resource profiles describing the available resources and the processing profile describing the resource requirements of a process.

This simple mechanism used in DQS for job submission (queue complexes, described in chapter 2) is quite robust, and offers a high degree of flexibility. The limitation of the mechanism is that it doesn’t take into account the actual state of the resources and the resources are considered static e.g. if a background process is actually using most of the available memory within a machine the queuing services aren’t aware of this and will consequently start the process. Also the static description of resource attributes allows the specification of how much disk is available in total, but takes no account of the amount of data on the disk at any time. These limitations are very common, and arise from the lack of support for resource management in a UNIX environment. However, other solutions are available and clearly resource management is an issue that has to be addressed more thoroughly in future studies.

Job limits, status , resource reporting and checkpointing are all possible in DQS. Resource scheduling is not.

5.3.2 Software Criteria

DQS is available on a large number of workstation platforms, since there is a large user community for the product, and since the software is essentially free and distributed with the source the number of platforms being supported expands as new platforms are introduced.

DQS supports parallel execution of jobs on resources within the workstation cluster.

Duplicate master services can be run to avoid the single point of failure. Jobs which are executing when one of the machines crash are lost, although the operator and originator of these jobs are informed.

DQS contains an GUI interface to the resources, although this interface was not found to be particularly well implemented or suitable. There are plans to generate a better interface. No API interfaces are available for the queuing services.

5.3.3 Other Criteria

Security and Accounting facilities are available within DQS, these have not been thoroughly evaluated as part of this prototype since it is anticipated that these features will be supplied by the management services in the final solution.

There is no facility for future planning of processing within the package.

5.3.4 Summary

In summary DQS is ideal in a smaller processing environment especially since it is relatively light weight (no reliance on DBMS etc.) DQS could be used as a final solution for the queuing services at a DAAC with a number of modifications, however it is anticipated that a more complete set of services would be available from other COTS packages.

Prototyping with DQS has been useful since it gives a good indication of the capabilities of a whole class of queuing packages. The software source is available and so detailed investigations of the internal workings of this type of package has been possible. There is an initiative to evaluate other COTS within the PDPS group, the experience from this prototype will be used in that evaluation. The later phase may adopt another set of queuing services depending on the precise aims of the prototype at that time.

6. Prototype Issues

6.1 Introduction

A number of technical issues have arisen during the data processing prototype which either have been addressed, or deferred to a later phase of the prototype. In general these issues are both pertinent to the prototype and the final system. These issues are presented along with the current prototype approach / assumptions for the issue along with any plans for further investigation.

6.2 Issue: Toolkit Interface via Processing Script

Description: The SDP toolkit takes as input Process Control information in the form of a PC table. This forms the main interface to the PGE from the Data Processing subsystem. A simple approach to providing this information to the toolkit, from the processing environment has been implemented, this approach should be evaluated for adoption in PDPS design.

Current Approach: A processing script is used to encapsulate a PGE. Within this script PC table information is defined and written to a file, the PC info file, when the script is executed. The location of the file is also defined in the script, and set up as an environmental variable, which the toolkit reads and uses to access the file. This method has been used to interface to both the AVHRR pathfinder algorithm and the SSMI pathfinder, and is effective. The form of the PC table entries is such that some items need to be predefined in the script template, e.g.. the logical numbers of the items. An example template entry for an item in the PC file is shown here

```
echo "402|<@INPUT_FILE2=@>|<@INPUT_DIR=@>||||1" >> $PGS_PC_INFO_FILE
```

where the token <@INPUT_FILE2=@> is replaced by a filename when the script is qualified. The variable PGS_PC_INFO_FILE has been set up to contain the PC info filename . Hence the data

```
402|{filename}|{directory}||||1
```

is written into the PC info file in this example, and read by the Toolkit.

The approach used is quite generic and can be used to supply other information required in the processing script such as environmental variables and resource requirements.

Approach/Follow Up: The method currently in use is sufficient to communicate the information to the SDP Toolkit. The use of a flat file as the medium allows the storage of this information in the data server as log information but is not necessarily the most efficient technique. It may be necessary at a later date to communicate to the Toolkit via a database instead. It is anticipated that the processing script can still be used to transfer the information in that scenario.

Target Schedule: No further work will be carried out on the Toolkit Interface, unless particular issues are raised through the integration of further scientific algorithms or through major changes to the Toolkit design.

6.3 Issue: Data Processing Request Information

Description: The Data Processing subsystem is responsible for managing, queuing and executing Data Processing Requests. These will be sent by the planning subsystem Production Management Service.

Current Approach: Data Processing Requests are entered via a GUI in the Phase 1 prototype. Information contained in the request is substituted into the processing scripts that encapsulate the PGEs, and hence is passed on to the PGE via the PC table and environmental variables and also to the process queuing service. A GUI is provided to enable validation of DPRs by the operator.

The information within the DPR is expected to contain:

- Unique Run Identifier
- Input data identifiers: granule id, granule size and staging method -- it is anticipated that this information may be presented as a URL for the data
- Output data identifiers: granule id and anticipated size
- Processing parameters
- Resource requirements: either specific information of precise processing resources, disks to be used, or a more generic specification of the class of resources required
- Environmental variable values (as required by PGE)

The prototype allows the use of default values if information is not contained in the DPR.

Currently, the operator initiates chains of PGEs that are described in pre-defined templates, rather than individual DPEs, and then validates/edits parameters for the individual PGEs.

Approach/Follow Up: An interface layer needs adding to the CHAINS software to allow API-level input of Data Processing Requests. The software used to process the requests has been designed in such a way as to provide APIs that may be used by that interface layer. The precise definition of DPRs will need defining in collaboration with the Planning subsystem.

Target Schedule: Phase 3 of the prototype involves interfacing to the Planning subsystem prototype. As part of this process the remaining issues over DPRs will be resolved.

6.4 Issue: Resource Model and Resource Management

Description: There are a number of alternate approaches to resource management. The data processing prototype uses a simple abstract model of resources to base scheduling decisions to put jobs on suitable machines (also using load usage statistics for the resources to perform simple load balancing). The resource management task is one of the more complicated tasks within the data processing environment.

Current Approach: The simple resource model within the prototype was described in chapter 3, the limitations of this resource model are that the actual usage of the resources is not accounted for.

Follow Up: It is proposed to study the resource management capabilities of other COTS queuing packages, as well as obtaining a better understanding of what might be provided by the management infrastructure. In addition to this, there are a interesting technologies emerging for better management of resources from UNIX, such as in the Andrew File System (AFS) for reserving disk space resources. The way that these technologies could impact the design of the resource management task will be investigated.

Target Schedule: It is proposed that during Phase 2 these issues are studied further, with the anticipation that further prototyping may be required during Phase 3.

6.5 Issue: Data Staging / Predictive Staging

Description: The SDS anticipates the need for predictive staging of data in the data processing sub-system. The needs for predictive staging are still unclear, modeling and prototype results will be required to validate the requirements. There are a number of alternatives for staging management: from simple FIFO queues which ensure that the data are available for as long as possible on the local disks, to more complex forward looking capabilities which anticipate which data are required in the near future. Although predictive staging is unlikely to be required for early releases of the ECS system the impact that this could have on the system design should be understood at an earlier stage.

Current Approach: Within the current prototype data are staged as part of the processing script. Data staging is currently achieved via various file transfer commands (ftp, cp, rcp etc.) within a processing script.

Follow Up: It is proposed to investigate the different possibilities / architecture for data staging. There are clearly interface issues with the resource management and data server sub-systems.

Schedule: It is proposed that within Phase 2 of the prototype these issues are studied, with the anticipation of providing simulation / prototype components for data staging from archive in the Phase 3 time frame.

6.6 Issue : QA

Description: The Data Processing environment should be capable of supporting QA operations.

Current Approach: Three categories of QA operation have currently been identified (ref. : PDPS telecon, 11/18/94). These are

- In line QA	- automated QA.
- SCF QA	- off-line, at another site to the DPS
- DAAC manual QA	- human-in-the-loop QA local to the DAAC.

The current design concept allows in-line QA to be handled as an ordinary PGE. However, it is not clear whether SCF QA and DAAC manual QA should be handled this way. During Phase 1 of the prototype work was brought forward to verify that the X-display aspects of manual QA could be handled by the Data Processing prototype.

Approach / Follow Up: QA operations may well be carried out both independently of the product processing chains and in-line to the chain. In both cases it is likely that they will add a degree of conditionally to the control flows through the PGEs The Data Processing prototype should address these conditionally aspects, which may also apply to other types of PGEs. In addition it should address the resource assignment aspects relating to manual QA processes, which may be of indeterminate duration. Furthermore, it should look at how SCF QA is modeled.

Target Schedule: This issue will be addressed during Phase 2 of the Data Processing prototype.

6.7 Issue : Script Languages

There are a number of advanced scripting languages gaining a large popularity under UNIX, e.g. Perl, Python Tcl. These are being used increasingly in developing high level applications and may be requested by science groups for developing their algorithms. In addition, locally developed scripting languages may be in use.

Current approach: Scripts are currently assumed to be UNIX.

Approach / Follow Up: There are a number of related issues that need should be addressed

- the level of support required from the processing environment to permit use of these scripts
- toolkit compatibility issues
- what benefits the PDPS group could get from adopting one of these scripting languages within the system design

Target Schedule: It is proposed that these issues will be addressed during Phase 2 of the DP prototype.

6.8 Issue : Process Statusing

Description: Process status information must be available to the Data Processing subsystem operator for monitoring purposes and also to external subsystems (such as the planning subsystem) for progress monitoring.

Current Approach: Basic status information is gathered by continuous monitoring of queues and processing resources which form the Process Queuing Service and Process Execution Service. This provides information at a state level - i.e. unstarted, queued, processing, complete, held, error or skipped, which is displayed on the operator GUI of the DP prototype.

Approach / Follow Up: The issues that should be addressed are

- methods for gaining more detailed status information
- use of status information for conditionally controlling the process flow
- toolkit issues relating to the need for status information
- provision of status information to other subsystems

- possibilities for use of status information within resource monitoring, scheduling and staging

Target Schedule: It is proposed to address these issues during phase 2 of the Data Processing prototype.

6.9 Issue: Checkpointing

Description: Checkpointing is the ability to save processing results at intermediate points in the processing cycle. Checkpointing is used for many purposes,

- to save intermediate results to permit restart from a further point in the processing
- to permit an algorithm to close down in an ordered close down
- new technologies are investigating architecture independent checkpointing the ability to move processing from one machine to another (migratory load balancing).

Current Approach: Checkpointing has not been addressed in the prototype so far

Follow up: It is proposed to study the potential impact that supporting this type of feature would have on the PDPS design

Target Schedule: Phase 3

6.10 Issue: Science Software Execution Issues

Description: What extra support is required from the queuing services for non-sequential processing paradigms

Current Approach: DQS supports concurrent scheduling of algorithms on distributed workstations.

Follow Up: A more detailed investigation is required to understand the impact of all the processing paradigms within the science software execution prototype impact the requirements / design of the PDPS. These impacts may be in terms of functions within the queuing / execution service, or interfaces to packages required to support the processing paradigm (e.g. MPI, PVM, DCE etc.), as well as impacts on the resource management requirements.

Target Schedule: Phase 3

References

1. Science Software Execution Prototype: Report 2: AVHRR Pathfinder. Narayan Prasad (available via EDHS)

2. SDP Toolkit Implementation With Pathfinder SSM/I Precipitation Rate Algorithm. Narayan Prasad (available via EDHS)

3. AVHRR Pathfinder Home Page (URL: http://www.xtreme.gsfc.nasa.gov/)

4. DQS Home Page (URL: http://www.scri.fsu.edu/~pasko/dqs.html)

5. A Comparison of Queuing, Cluster and Distributed Computing Systems - NASA Technical Memorandum 109025 (revision 1) June 1994. J Kaplan and M Nelson.

6. Draft Standard for Information Technology - POSIX - Part 2: Shell and Utilities -- Amendment: Batch Environment. IEEE P1003.2d / D12.2

7. Portable Batch System External Reference Specification: R Henderson et. al NAS Systems Development Branch - NASA Ames Research Center

8. Data Processing Prototype - Preliminary Design Document. Jolyon Martin (available via EDHS)